Plans for Migrating SEAL Functionality

- Current Status
- Proposed actions
- REFLEX/PluginService functionality

P. Mato / CERN

Remaining SEAL Functionality

- The functionality that is left from the SEAL project is the following:
  - **Foundation** functionality implemented by the packages: *SealBase*, *SealIOTools*, *SealZip* and *SealUtil*
  - **PluginManager** and related utility programs (*SealPluginDump*, etc.)
  - **Component Model** implemented by the packages: *SealKernel* and *SealServices*.
  - **MathLibs**, which still includes FML (fit and minimization) package

Usages Assumptions

- Foundation
  - These packages are mainly used by CMS directly, and also at some level by the Persistency Framework projects (CORAL, COOL, POOL). Indirectly are also used by LHCb and ATLAS.
- Plugin Manager
  - The SEAL Plugin manager is used by the Persistency Framework project and CMS. The new CMS framework CMSSW relies heavily on it.
- Component Model
  - The two packages of the component model are used exclusively by the Persistency Framework projects. No use in CMS has been reported. Recently, some modifications had to be done to better support multi-threading.

Proposed Actions

- The subset component model functionality from *SealKernel* and *SealServices* which is used by the persistency framework will be moved to the CORAL project
- The dependencies of these packages to SealBase will be eliminated.
- The moved component model to CORAL will be for internal use of the Persistency Framework packages.
- Two options for the replacement of the Plugin Manager in persistency framework packages
  a) Re-implement the *ComponentLoader* with a set of "hardwired" relations between Persistency components and the name of the library that needs to be loaded and the factories to be located
  b) The new class (PluginService) available already in ROOT/Reflex will be used to provide plugin management functionality

Proposed Actions (2)

- The SEAL plugin manager will only be used by CMS and could be moved into their code base.
- Move the remaining SEAL packages: SealBase, SealIOTools and SealZip or portions of them to the CMS code base.

- ✓ The time scale for CORAL changes could be by the end-January for a development release and by end-February for a production release
- ✓ CMS will have to decide for their actions and time scale. Meanwhile SEAL will continue to be released.
- ✓ AF endorsed the proposal

Reflex Plugin Service

- This package has been developed to enhance and simply some aspects of the GAUDI framework
- The main goals have been
  - Remove the need for the property "ApplicationMgr.Dlls". Component libraries could be loaded on demand
  - Simplification of the code. Replace existing factories (AlgFactory, ToolFactory, etc.) with a single method, so many classes can be removed from Gaudi
  - Compatibility with other plugins and dictionary systems since they are based also on roopmap files
  - Dependent exclusively on the Reflex package, such that can be added into ROOT/Reflex package
  - Possible replacement for the SEAL plugin manager that could be of interest for CORAL, POOL, COOL, etc.

Using Plugin Service

- Coding the plugin/component
  - No predefined model
  - Declaring factory with signature

```
class MyClass : public ICommon {
  MyClass(int, ISvc*);
  ...
};
```

MyClass.h

```
PLUGINSVC_FACTORY(MyClass,ICommon*(int,ISvc*));
/* implementation */
```

MyClass.cpp

- Creating the rootmap file
  - Text file listing all plugins and the associated dynamic library
  - The build system creates it with *genmap*

```
Library.MyClass:          MyLibrary.so
Library.AnotherClass:     MyLibrary.so
```

rootmap

- Instantiating the plugin
  - Library loaded if needed
  - Strong argument type checking

```
...
ISvc* svc = ...
ICommon* myc;
myc = PluginSvc::create<ICommon*>("MyClass",10, svc);
if ( myc ) {
  myc->doSomething();
}
```

Program.cpp

New Plugin Service

- Standalone package with a single dependency to Reflex
- Implementation:

```
  l     w     c
112   411  3333 PluginSvc/src/lib/FactoryMap.cxx
163   664  5727 PluginSvc/src/lib/PluginSvc.cxx
108   389  3071 PluginSvc/src/lib/SharedLibrary.cxx


112   318  2472 PluginSvc/PluginSvc/dirmanip.h
 44   151  1252 PluginSvc/PluginSvc/FactoryMap.h
200   980  9724 PluginSvc/PluginSvc/PluginSvc.h
 20    89   653 PluginSvc/PluginSvc/SharedLibrary.h
```

Example: integration in GAUDI

- Redefinition of a number of macros

```
#define DECLARE_ALGORITHM_FACTORY(x) \
     PLUGINSVC_FACTORY(x,IAlgorithm*(std::string, ISvcLocator*))

#define DECLARE_SERVICE_FACTORY(x) \
     PLUGINSVC_FACTORY(x,IService*(std::string, ISvcLocator*))

#define DECLARE_TOOL_FACTORY(x) \
     PLUGINSVC_FACTORY(x,IAlgTool*(std::string, std::string, ISvcLocator*))


#define DECLARE_CONVERTER_FACTORY(x) \
   PLUGINSVC_FACTORY_WITH_ID( x, ConverterID(x::storageType(),x::classID()),\
                              IConverter*(ISvcLocator*))
```